

Optimisation of Quantum Evolution Algorithms

Apoorva Patel*

CHEP and SERC, Indian Institute of Science, Bangalore 560012, India

E-mail: adpatel@cts.iisc.ernet.in

Given a quantum Hamiltonian and its evolution time, the corresponding unitary evolution operator can be constructed in many different ways, corresponding to different trajectories between the desired end-points. A choice among these trajectories can then be made to obtain the best computational complexity and control over errors. As an explicit example, Grover's quantum search algorithm is described as a Hamiltonian evolution problem. It is shown that the computational complexity has a power-law dependence on error when a straightforward Lie-Trotter discretisation formula is used, and it becomes logarithmic in error when reflection operators are used. The exponential change in error control is striking, and can be used to improve many importance sampling methods. The key concept is to make the evolution steps as large as possible while obeying the constraints of the problem. In particular, we can understand why overrelaxation algorithms are superior to small step size algorithms.

The 32nd International Symposium on Lattice Field Theory

23-28 June 2014

Columbia University, New York, NY

*Speaker.

Classical computer simulations of quantum systems are not efficient—well-known examples range from the Hubbard model to lattice QCD—and Feynman argued that quantum simulations would do far better [1]. The essence of the argument is that quantum simulations sum multiple evolutionary paths (in superposition) contributing to a quantum process at one go, while classical simulations evaluate these paths one by one. Formalisation of this advantage for simulating physical Hamiltonians, in terms of computational complexity, has improved over the years step by step [2, 3, 4, 5, 6]. Here, treating Grover’s quantum search algorithm as a Hamiltonian evolution problem, we expose the physical reasons behind the improvement in computational complexity.

Computational complexity of a problem is a measure of the resources needed to solve it. Conventionally, the computational complexity of a decision problem is specified in terms of the size of its input, noting that the size of its output is only one bit. Problems with different output requirements are reduced to a sequence of decision problems, with gradually narrowing bounds on the output adding one bit of precision for every decision made. In such a scenario, the number of decision problems solved equals the number of output bits, and it is appropriate to specify the complexity of the original problem in terms of the size of its input as well as its output. Generalising the conventional classification, the computational algorithm then can be labeled efficient if the required resources are polynomial in terms of the size of both its input and its output.

Popular importance sampling methods are not efficient according to our criterion, because the number of iterations needed in the computational effort has a negative power-law dependence on the precision ε (i.e. $N_{\text{iter}} \propto \varepsilon^{-2}$ as per the central limit theorem). On the other hand, finding zeroes of a function by bisection is efficient (i.e. $N_{\text{iter}} \propto \log \varepsilon$), and finding them by Newton’s method is super-efficient (i.e. $N_{\text{iter}} \propto \log \log \varepsilon$).

1. Quantum Hamiltonian Simulation

The Hamiltonian simulation problem is to evolve an initial quantum state $|\psi(0)\rangle$ to a final quantum state $|\psi(T)\rangle$, in presence of interactions specified by a Hamiltonian $H(t)$:

$$|\psi(T)\rangle = U(T)|\psi(0)\rangle, \quad U(T) = P\left[\exp\left(-i \int_0^T H(t)dt\right)\right]. \quad (1.1)$$

Alternatively, the problem can be defined as determination of the evolution operator $U(T)$, without any mention of the initial and the final states. The norm of the difference between the simulated and the exact evolution operators specifies the simulation accuracy, say $\|\tilde{U}(T) - U(T)\| < \varepsilon$.

We restrict ourselves here to Hamiltonians acting in finite N -dimensional Hilbert spaces. A general $H(t)$ then be a dense $N \times N$ matrix, and there is no efficient way to simulate it. So we furthermore assume that $H(t)$ the following features commonly present in physical problems:

- (1) The Hilbert space is a tensor product of many components, e.g. $N = 2^m$ for a system of qubits.
- (2) The components have only local interactions irrespective of the size of the system, e.g. only nearest neighbour couplings. That makes $H(t)$ sparse, with $O(N)$ non-zero elements.
- (3) $H(t)$ is specified in terms of a finite number of functions, while the arguments of the functions can depend on the components, e.g. the interactions are translationally invariant. That allows $H(t)$ to have a compact description, and consequently the resources needed to just write down $H(t)$ do not influence the simulation complexity.

Such Hamiltonians can be mapped to graphs with bounded degree d , with vertices \leftrightarrow components and edges \leftrightarrow interactions. Their simulations can be easily parallelised—on classical computers, they allow SIMD simulations with domain decomposition. With these criteria, efficient Hamiltonian simulation algorithms are those that use resources polynomial in $\log(N)$, d and $\log(\epsilon)$.

1.1 Hamiltonian Decomposition

Efficient simulation strategy for Hamiltonian evolution has two major ingredients. The first ingredient is to decompose the sparse Hamiltonian as a sum of non-commuting but block-diagonal Hermitian operators, i.e. $H = \sum_{i=1}^l H_i$. Then each H_i can be easily and exactly exponentiated for any time evolution τ , with $\exp(-iH_i\tau)$ retaining the same block-diagonal structure. Reducing the block size all the way to 2×2 , the blocks become linear combinations of projection operators. Projection operators with only two distinct eigenvalues can be interpreted as binary query oracles.

In general, H_i can be identified by an edge-colouring algorithm for graphs [3], with distinct colours for overlapping edges. At most $d + 1$ colours are needed to efficiently colour any sparse graph. Identification of H_i also provides a compressed labeling scheme that can be used to address individual blocks. The number of blocks is $O(m) = O(\log N)$, and they can be evolved simultaneously, in parallel (classically) or in superposition (quantum mechanically).

For example, even and odd edges of a linear chain provide a block-diagonal decomposition of the one-dimensional Laplacian operator, $H = H_o + H_e$. Its projection operator structure follows from $H_o^2 = 2H_o$ and $H_e^2 = 2H_e$. The last bit of the position label identifies H_o and H_e . Eigenvalues of H are $4\sin^2(k/2)$ in terms of the lattice momentum k , while those of H_o and H_e are just 0 and 2.

1.2 Evolution Optimisation

Given that individual H_i can be exponentiated exactly and efficiently, their sum H can be approximately but efficiently exponentiated using the discrete Lie-Trotter formula:

$$\exp(-iHT) = \exp\left(-i\sum_i H_i T\right) \approx \left(\prod_i \exp(-iH_i \Delta t)\right)^n, \quad n = T/\Delta t. \quad (1.2)$$

This approximation retains unitarity of the evolution, but may not preserve other properties such as the energy. The accuracy of the approximation is usually improved by decreasing Δt . This method has been used in classical parallel computer simulations of quantum evolution problems [7, 8].

In contrast, the second ingredient of efficient Hamiltonian simulation is to use as large Δt as possible. When the exponent is proportional to a projection operator, the largest Δt is the one that makes the exponential a reflection operator. Such an extreme strategy not only keeps the evolution accurate but also improves the algorithmic complexity from a power-law dependence on ϵ to a logarithmic one. This is not obvious, and we demonstrate it next for the quantum search problem.

2. Quantum Search as Hamiltonian Evolution

The quantum search algorithm works in an N -dimensional Hilbert space, whose basis vectors $\{|i\rangle\}$ are identified with the individual items. It takes the initial state $|s\rangle$ whose amplitudes are uniformly distributed over all the items, to the target state $|t\rangle$ where all but one amplitudes vanish.

$$|\psi(0)\rangle = |s\rangle, \quad |\psi(T)\rangle = |t\rangle, \quad |\langle i|s\rangle| = 1/\sqrt{N}, \quad \langle i|t\rangle = \delta_{it}. \quad (2.1)$$

The simplest evolution schemes taking $|s\rangle$ to $|t\rangle$ are governed by time-independent Hamiltonians that depend only on $|s\rangle$ and $|t\rangle$. The unitary evolution is then a rotation at a fixed rate in the two-dimensional subspace, formed by $|s\rangle$ and $|t\rangle$, of the whole Hilbert space. In this subspace, let

$$|t\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |t_\perp\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad |s\rangle = \begin{pmatrix} 1/\sqrt{N} \\ \sqrt{(N-1)/N} \end{pmatrix}. \quad (2.2)$$

There are many evolution routes with $U(T)|s\rangle = |t\rangle$, and we consider two particular cases in turn.

2.1 Farhi-Gutmann's and Grover's Algorithms

Grover based his algorithm on a physical intuition for the Hamiltonian [9], where the potential energy term $|t\rangle\langle t|$ attracts the wavefunction towards the target state and the kinetic energy term $|s\rangle\langle s|$ diffuses the wavefunction over the whole Hilbert space. Both the terms are projection operators, and the time-independent Hamiltonian is

$$H_C = |s\rangle\langle s| + |t\rangle\langle t| = I + \frac{\sqrt{N-1}}{N}\sigma_1 + \frac{1}{N}\sigma_3. \quad (2.3)$$

The corresponding evolution operator is (without the global phase)

$$U_C(t) = \exp(-i\hat{n} \cdot \vec{\sigma} t/\sqrt{N}), \quad \hat{n} = (\sqrt{(N-1)/N}, 0, 1/\sqrt{N})^T, \quad (2.4)$$

which is a rotation by angle $2t/\sqrt{N}$ around the direction defined by \hat{n} on the Bloch sphere.

The (unnormalised) eigenvectors of H_C are $|s\rangle \pm |t\rangle$. They correspond to the directions $\pm\hat{n}$, and bisect the initial and the target states. Thus a rotation by angle π around \hat{n} takes $|s\rangle\langle s|$ to $|t\rangle\langle t|$ on the Bloch sphere, and the time required for the Hamiltonian search is $T = (\pi/2)\sqrt{N}$ [10].

Grover made an enlightened jump from this scenario, motivated by the Lie-Trotter formula. He exponentiated the projection operators in H_C to reflection operators; $R = \exp(\pm i\pi P) = 1 - 2P$ for any projection operator P . His optimal algorithm iterates the discrete evolution operator [11],

$$U_G = -(1 - 2|s\rangle\langle s|)(1 - 2|t\rangle\langle t|) = (1 - \frac{2}{N})I + 2i\frac{\sqrt{N-1}}{N}\sigma_2. \quad (2.5)$$

With $U_G = \exp(-iH_G\tau)$, it corresponds to the Hamiltonian and the evolution step:

$$H_G = \frac{i}{\sqrt{N}}(|t\rangle\langle s| - |s\rangle\langle t|) = i[|t\rangle\langle t|, |s\rangle\langle s|] = -\frac{\sqrt{N-1}}{N}\sigma_2, \quad \tau = \frac{2N}{\sqrt{N-1}}\sin^{-1}\left(\frac{1}{\sqrt{N}}\right). \quad (2.6)$$

It is an important non-trivial fact that H_G is the commutator of the two projection operators in H_C .

On the Bloch sphere, each U_G step is a rotation by angle $2\tau\sqrt{N-1}/N = 4\sin^{-1}(1/\sqrt{N})$ around the direction $\hat{n}_G = (0, 1, 0)^T$, taking the geodesic route from the initial to the final state. That makes the number of steps required for this discrete Hamiltonian search,

$$Q_T = \frac{\cos^{-1}(1/\sqrt{N})}{2\sin^{-1}(1/\sqrt{N})} \approx \frac{\pi}{4}\sqrt{N}. \quad (2.7)$$

Note that \hat{n} and \hat{n}_G are orthogonal, so the evolution trajectories produced by rotations around them are completely different, as illustrated in Fig.1. It is only after a specific evolution time, corresponding to the solution of the quantum search problem, that the two trajectories meet.

To compare the rates of these two Hamiltonian evolutions, we observe that H_C can be simulated by alternating small evolution steps governed by $|s\rangle\langle s|$ and $|t\rangle\langle t|$, according to the Lie-Trotter formula. Then each evolution step governed by $|t\rangle\langle t|$ needs two binary queries [12]. On the other hand, U_G can be simulated using only one binary query per evolution step.

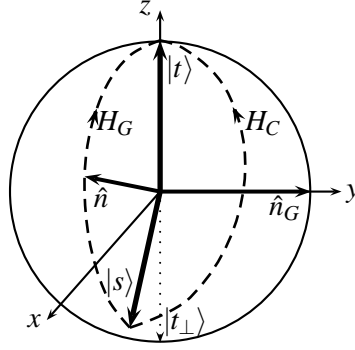


Figure 1: Evolution trajectories on the Bloch sphere for the quantum search problem, going from $|s\rangle$ to $|t\rangle$. The Hamiltonians H_C and H_G generate rotations around the directions \hat{n} and \hat{n}_G respectively.

2.2 Equivalent Evolutions

Two Hamiltonian evolutions are truly equivalent, when their corresponding unitary evolution operators are the same (upto a global phase). The intersection of the two evolution trajectories is then independent of the specific initial and final states. For the quantum search problem, we find

$$U_C(T) = i(1 - 2|t\rangle\langle t|) (U_G)^{Q_T} . \quad (2.8)$$

For a general evolution time $0 < t < T$, we have the relation (similar to Euler angle decomposition),

$$U_C(t) = \exp(i\beta\sigma_3) (U_G)^{Q_t} \exp\left(i\left(\frac{\pi}{2} + \beta\right)\sigma_3\right) , \quad (2.9)$$

i.e. $U_C(t)$ can be generated as Q_t iterations of the Grover operator U_G , preceded and followed by phase rotations. Here $\sigma_3 = 2|t\rangle\langle t| - 1$ is a known reflection, and

$$Q_t = \frac{\sin^{-1}\left(\sqrt{\frac{N-1}{N}} \sin(t/\sqrt{N})\right)}{2\sin^{-1}(1/\sqrt{N})} \approx \frac{t}{2} , \quad \beta = -\frac{\pi}{4} - \frac{1}{2}\tan^{-1}\left(\frac{1}{\sqrt{N}} \tan(t/\sqrt{N})\right) . \quad (2.10)$$

It is truly remarkable that H_G can be used to obtain the same evolution as H_C , even though the two Hamiltonians are entirely different in terms of their eigenvectors and eigenvalues!

2.3 Discretised Hamiltonian Evolution Complexity

Digitisation of continuous variables is necessary for fault-tolerant computation with control over bounded errors. But it also introduces discretisation errors that must be kept within specified bounds. The algorithmic error of the Lie-Trotter formula depends on Δt , which has to be chosen so as to satisfy the total error bound ε on $U(t)$. For the simplest discretisation scheme,

$$\exp\left(-i\sum_{i=1}^l H_i \Delta t\right) = \exp(-iH_1 \Delta t) \dots \exp(-iH_l \Delta t) \times \exp(-iE^{(2)}(\Delta t)^2) , \quad (2.11)$$

$$E^{(2)} = \frac{i}{2} \sum_{i < j} [H_i, H_j] + O(\Delta t) . \quad (2.12)$$

For unitary operators X and Y , Cauchy-Schwarz and triangle inequalities give,

$$\|X^n - Y^n\| = \|(X - Y)(X^{n-1} + \dots + Y^{n-1})\| \leq n\|X - Y\| . \quad (2.13)$$

So for the total evolution to remain within the error bound ε , we need

$$n \|\exp(-iE^{(2)}(\Delta t)^2) - I\| \approx n \|E^{(2)}\| (\Delta t)^2 = t \|E^{(2)}\| (\Delta t) < \varepsilon. \quad (2.14)$$

With exact exponentiation of the individual terms H_i , the computational cost to simulate a single time step Δt , \mathcal{C} , does not depend on Δt . The complexity of the Hamiltonian evolution is then

$$O(n\mathcal{C}) = O\left(t^2 \left(\frac{\|E^{(2)}\|}{\varepsilon}\right) \mathcal{C}\right). \quad (2.15)$$

With superlinear scaling in t and power-law scaling in ε , this small Δt scheme is not efficient.

Grover's optimal algorithm uses a discretisation formula where $\exp(-iH_i\Delta t_G)$ are reflection operators. The corresponding time step is large, i.e. $\Delta t_G = \pi$ for Eq.(1.2) applied to Eq.(2.3). The large time step introduces an error because one may jump across the target state during evolution instead of reaching it exactly. Q_t is not an integer as defined in Eq.(2.10), and needs to be replaced by its nearest integer approximation $\lfloor Q_t + \frac{1}{2} \rfloor$ in practice. Since each time step provides a rotation by angle $\alpha = 2\sin^{-1}(1/\sqrt{N})$, and one may miss the target state by at most half a rotation step, the error probability of Grover's algorithm is bounded by $\sin^2(\alpha/2) = 1/N$, independent of the number of time steps. Since the preceding and following phase rotations in Eq.(2.9) are unitary operations, this error bound applies to $U_C(t)$ as well. Thereafter, multiple runs of the algorithm and selection of the result by majority rule can rapidly reduce the error probability. With R runs, the error probability becomes less than $2^{R-1}/N^{\lceil R/2 \rceil}$, which can be made smaller than any prescribed error bound ε . (In a drastic contrast, averaging the results of multiple runs would make the error probability smaller than $1/(N\sqrt{R})$ only.) The computational complexity of the evolution is thus

$$O(Q_t R \mathcal{C}_G) = O\left(\frac{t}{2} \left(-\frac{2\log \varepsilon}{\log N}\right) \mathcal{C}_G\right) = O\left(-t \frac{\log \varepsilon}{\log N} \mathcal{C}_G\right). \quad (2.16)$$

With linear scaling in time and logarithmic scaling in ε , this algorithm is efficient.

To complete the analysis, we note that a digital computer with a finite register size also produces truncation errors. With b -bit registers, the available precision is $\delta = 2^{-b}$. With all functions approximated by accurate polynomials, and Euler angle decomposition reducing rotations about arbitrary axes to rotations about fixed axes, an individual H_i can be exponentiated to b -bit precision with $O(mb^3)$ effort. The number of exponentiations of H_i needed for the Lie-Trotter formula is nl , which reduces to $2Q_t$ for the Grover version. So with the choice $nl\delta = O(\varepsilon)$, i.e. $b = \Theta(\log(n/\varepsilon))$, the truncation error becomes negligible compared to the discretisation error. The cost of a single evolution step then scales as $\mathcal{C} = O(m(\log(t/\varepsilon))^3)$, which is efficient.

3. Extensions and Outlook

It is straightforward to extend the preceding results to other Hamiltonians consisting of only two projection operators, e.g. the staggered Dirac operator for free fermions in any number of dimensions [13]. For Hamiltonians that are linear combinations of more than two projection operators, e.g. three projection operators for the discretised Laplacian on the graphene lattice, successive H_i can be added to the algorithm one by one in an inductive procedure. The resultant large Δt evolution is not exact, but it still has $\Theta(1)$ success probability for a suitable choice of Δt . That keeps the overall scaling of the evolution efficient, $O(lt\|H\|\log(lt\|H\|/\varepsilon)\mathcal{C})$ [5, 6].

Our construction of the efficient Hamiltonian evolution algorithm relies on: (1) simplification of the Baker-Campbell-Hausdorff expansion for products of exponentials of projection operators, and (2) conversion of the results to a digital form allowing selection of the best one by majority rule. These algebraic properties are not specific to quantum computers; they can be incorporated in and readily benefit traditional classical simulations of quantum systems. In particular:

- (a) It is known that overrelaxation algorithms [14], based on evolution steps that are reflections consistent with conservation laws, provide a much more efficient sampling of the configuration space (measured in terms of the autocorrelation time) than the small step size Metropolis algorithm. The analysis presented here provides an understanding of that observation.
- (b) Many physical problems with periodic patterns are solved using the fast Fourier transform. The block-diagonal decomposition provides a competitive real space method for solving them.
- (c) Projection operator decomposition of the Hamiltonian can be easily found for quantum Monte Carlo problems, and for molecular dynamics simulations using the Lie-Trotter formula in Euclidean time. The remaining task is to find a useful large step size with high acceptance probability.
- (d) Evaluations of functions other than exponentials may also simplify with the block-diagonal projection operator decomposition. A case of particular interest is the evaluation of the fermion determinant appearing in many physical problems.

Work on such applications is in progress.

References

- [1] R.P. Feynman, Int. J. Theor. Phys. 21 (1982) 467.
- [2] S. Lloyd, Science 273 (1996) 1073.
- [3] D. Aharonov and A. Ta-Shma, Proc. 35th Annual ACM Symp. on Theory of Computing (STOC'03), San Diego (2003), p.20 [arXiv:quant-ph/0301023].
- [4] D.W. Berry, G. Ahokas, R. Cleve and B.C. Sanders, Comm. Math. Phys. 270 (2007) 359 [arXiv:quant-ph/0508139].
- [5] D.W. Berry, A.M. Childs, R. Cleve, R. Kothari and R.D. Somma, Proc. 46th Annual ACM Symp. on Theory of Computing (STOC'14), New York (2014), p.283 [arXiv:1312.1414].
- [6] A. Patel and A. Priyadarsini, to appear.
- [7] H. De Raedt, Comp. Phys. Rep. 7 (1987) 1.
- [8] J.L. Richardson, Comp. Phys. Comm. 63 (1991) 84.
- [9] L. Grover, Pramana 56 (2001) 333 [arXiv:quant-ph/0109116].
- [10] E. Farhi and S. Gutmann, Phys. Rev. A57 (1998) 2403 [arXiv:quant-ph/9612026].
- [11] L.K. Grover, Proc. 28th Annual ACM Symposium on Theory of Computing (STOC'96), Philadelphia (1996), p.212 [arXiv:quant-ph/9605043].
- [12] See for example: M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information*, (Cambridge University Press, 2000), Section 6.2.
- [13] A. Patel and Md.A. Rahaman, Phys. Rev. A82 (2010) 032330 [arXiv:1003.0065].
- [14] S. Adler, Phys. Rev. D23 (1981) 2901;
M. Creutz, Phys. Rev. D36 (1987) 515;
F.R. Brown and T.J. Woch, Phys. Rev. Lett. 58 (1987) 2394.